Automated multi-day tracking of mice for the analysis of social behavior

Shay Ohayon, Ofer Avni, Adam L. Taylor, Roian Egnor and Pietro Perona

Supplementary materials

1. Background subtraction

The background model of the arena is automatically estimated by the system by sampling 50 evenly spaced video frames and computing their pixelwise median B (see Supplementary Fig. 1b). Subsequently, foreground pixels (F) of any frame I were defined as those that differ from the background (B) by a fixed amount:

$$\text{Eq1: } F = \left| I - B \right| > Th ,$$

where Th is the threshold. The foreground image is composed of all foreground pixels. The threshold is computed with the help of the user, who is prompted to place ellipses on the mice that are visible in 7 randomly selected frames. Using this information, the optimal threshold Th for background subtraction is computed by minimizing a cost function that counts false alarms (the number of pixels outside known mice positions) and misses (number of pixels that do not pass the threshold inside the known mice positions).

2. Tracking single mice

For each foreground image a morphological close operation (1mm) is applied to fill in missing pixels that do not exceed the thresholds (see Supplementary Fig. 1c-e). Small connected components are discarded and the remaining largest connected component (CC) is assumed to correspond to the mouse. An ellipse is fit to the largest connected component to approximate mouse shape (see Supplementary Fig. 1f and section below) . We call it the `mouse ellipse' in the following.

2.1 Fitting ellipses to connected components

The boundary of each connected component (CC) that is associated to a mouse is approximated in our system by an ellipse. Call $X_i = \left(x_i, y_i \right)$ the coordinates of the pixels in the CC; call μ and $\sum$ the mean and the covariance of the pixel coordinates. Then the ellipse is defined by the equation: $\left(X - \mu \right)^T \sum^{-1} \left(X - \mu \right) = 2^2$.

Notice that the ellipse is centered in μ, that the major and minor axes of the ellipse correspond to the eigenvectors of $\sum$ and that the with and length of the ellipse are equal to twice the square root of the eigenvalues of $\sum$.

An example of a fitted ellipse to foreground pixels is shown in Supplementary Figure 1f.

2.2 Collecting appearance exemplars

Exemplars of mice images are collected from the single-mouse training videos by sampling a rectangular patch tightly fitted around the ellipse outlining the mouse in each image. Pixels inside the patch are resampled using bi-linear interpolation, resulting in an 111x51 (10x5 mm) image patch showing the mouse in a standard orientation (i.e., head/tail facing towards the positive horizontal axis). Dense HOG features are extracted from the aligned image patch. We used block size of 10 pixels (3x9 blocks, 31 features per block), see (Felzenszwalb et al., 2010). This resulted in a feature vector of 837 dimensions. A small random subset of frames (~1000) is selected by the system and corresponding feature vectors are saved. Those feature values represent exemplars of known mice appearance.

2.3 Collecting head/tail exemplars

The ellipse that is fit to a mouse's image is ambiguous as to the animal's orientation. Mouse orientation may be estimated from its direction of motion when it is moving fast; when mice are moving slowly (or backward) such information is unreliable. Information on head/tail orientation may be obtained from the image as well. To train a head/tail classifier the system automatically collects exemplars of fast moving mice for which head orientation can be reliably determined based on velocity. For those frames, a bounding box is placed on the mouse ellipse and HOG features are computed on the aligned image patch (similar to appearance exemplars). These are used as positive exemplars for a mouse facing with its head to the right of the horizontal axes. The same image patches are then rotated 180 degrees and HOG features are computed for the rotated image patches. These features are used as negative exemplars (where tail is facing to the right of the horizontal axes). From these positive and negative examples a head-tail classifier is trained.

3. Tracking N mice

3.1 Parallel processing and jobs bootstraping

Video sequences are analyzed in parallel. The software for video recording automatically splits multiple-day recordings into 12-hour video files (~30GB each). Our system splits each video file into about 260 non-overlapping 5000-frame segments. Each video segment is analyzed independently of all other segments.

The analysis of each video segment starts by generating multiple hypotheses of mice positions and orientations for the first frame (see Supplementary Fig. 5a-b). The first frame is background subtracted and connected components are computed. Hypotheses are generated by computing all possible matches between connected components and N mice. Unlikely hypotheses, such as those containing ellipses that are too big (major axis larger than 55 pixels) or too small (major axis smaller than 18 pixels) are discarded.

Each hypothesis results in a tracking job with a different initial condition and is submitted to a computer cluster to be processed on one of the available nodes (see tracking algorithm below). The output of each job is N trajectories for the corresponding video segment, as propagated from the initial mice pose hypothesis in the first frame. The resulting video segment trajectories are then stitched together to obtain a final set of N trajectories for the entire video (see section below).

3.2 Tracking algorithm for a video with N mice

Tracking proceeds incrementally. Supppose that the position and orientation of the N mice has been computed at frame t-1 and frame t. The steps for analyzing frame t+1, are the following:

1. For each mouse, compute its predicted pose in frame t+1 by damped linear extrapolation of frames t and t-1: $p_{t+1}^i = p_t^i + d\left(p_t^i - p_{t-1}^i\right)$, where p_t^i corresponds to the i'th ellipse parameters at frame t (parameters are position, size and orientation). d is a damping coefficient used to smooth predictions. d typically equal 1, unless there was another mouse in close proximity in the previous frame (such that the two mice ellipse intersect). In the latter case, d is set to 0.1 which reduces spurious predictions due to possible poor segmentation.

2. Fit foreground pixels with a 2D Gaussian Mixture Model (N mixtures). Each mixture component corresponds to one of the mice. Fitting is done with Expectation Maximization algorithm (EM), see (Bishop, 2006). EM requires an initial solution and then iterates the mixture parameters until convergence. Multiple initial solutions (~M=15) are generated by perturbing the predicted ellipse positions with small random Gaussian noise.

3. Each of the M converged solutions contains N ellipses and is given a score to assess its goodness of fit to the actual image. The score estimates on how well the converged ellipses fit actual pixel values. This is done by sampling the image patch contained in a fixed bounding box that fits tightly around each ellipse I (see Supplementary Materials Sec. 2.2), computing the HOG features (Hi) of the image patch and comparing the features to the stored database of feature vectors (see section 2.2): $d_i = \min_Z \|H_i - Z\|$. d_i represents the minimal feature distance to a known mouse appearance, Z is the set of all stored feature vectors. The score of a solution is $\sum_{i=1}^{N} d_i$. The solution with the minimal score is selected and corresponds to the final ellipse placement in frame t+1.

4. Handle tracking failures / degenerate cases: if no detected pixels are found close to the placed ellipse in the previous 30 frames, consider this tracked mouse to be lost. Continue to track with N-1 mice.

5. If a large connected component appears that does not have an ellipse close to it and a mouse was previously lost, add an ellipse on newly detected connected component and declare the lost mouse found.

The process is then repeated for the next frame.

3.3 Merging jobs and stitching trajectories

To compute the mouse trajectory for the entire video sequence results from individual jobs need to be stitched together. Notice that some jobs analyzed the same video segment, but with different initial conditions. Therefore, the problem at hand is selecting the jobs with the correct initial conditions. Correct initial conditions will propagate well, while incorrect initial condition (say, two ellipses on the same mouse) will result degenerate events having unlabeled connected component.

Results from all jobs are stitched together using Dijkstra shortest path algorithm. The system constructs a directed acyclic graph (V,E), where V denotes the vertices and E denotes the edges (see Supplementary Fig. 5c). Each vertex v_i represents the tracked location of all four mice in a video segment. Two vertices v_i and v_j connect with an edge if the last frame of hypothesis v_i is the same as the first frame of hypothesis v_j (i.e. tree structure). Each edge is assigned a weight that is computed from two terms. The first term measures the similarity of mice poses in the last frame of job v_i to the first frame of job v_j (see Ellipse distance metric below). The second term counts how many

degenerate events occurred in job v_i. Degenerate events include mouse disappearing or reappearing. A large number of such events suggest the initial placement of ellipses was wrong. Once the graph is constructed, our system uses Dijstra's algorithm to compute the best stitching of the jobs trajectories into a quadruplet of video-length trajectories.

3.4 Correcting head/tail direction

The trajectories obtained from the tracker contain the position and orientation of N ellipses in each frame. However, the direction each mouse is still unknown (head/tail ambiguity). To solve for head/tail in each frame our system proceeds as described in supplementary section 2.3. The system solves the HMM using the Viterbi algorithm (Rabiner, 1989), which finds an optimal state sequence $S=\left[S_1,S_2,S_3,...,S_T\right]$ for the given observation sequence $O=\left[O_1,O_2,O_3,...,O_T\right]$. In this case, states correspond to orientation angles (360 states with 1 deg resolution). State S_i therefore can have 360 discrete states. Observation $\mathrm{O_i}$ includes the measured ellipse orientation (β), measured speed (v), measured velocity direction (θ) and pixel values (I) inside the fixed size ellipse bounding box.

We define the observation probability as two independent components:

$$\text{Eq2: } p\left(O_i \mid S_i\right)=p\left(v,\theta \mid S_i\right)p\left(\beta,I \mid S_i\right)$$

The multiplication in the likelihood term represents the convenient assumption that the observed velocity of a tracked ellipse is independent of the observed orientation and pixel values inside once the orientation of the mouse is known. Although this assumption may not be true for high velocities, it allows to simplify computations considerably and works well.

We model the first factor of the r.h.s of equation 1 as a von-Mises distribution with a spread κ that depends on the velocity v and its direction θ :

$$\text{Eq3: } p\left(v,\theta \mid S_i=\alpha\right)=\frac{e^{K_v\cos(\alpha-\theta)}}{2\pi I_0\left(K_v\right)},$$

where I_0 is the modified Bessel function of order 0.

Intuitively, the probability of observing a direction θ that is far away from the mouse's orientation α should be low. In practice, this depends on the instantaneous velocity. When velocity is high, we model the distribution as a narrow Gaussian around $\alpha-\theta$ and

use low kappa values for the spread, and when velocity is low, we model the distribution with a broader circular Gaussian, reflecting the uncertainty in θ. We model these concepts by using a spread parameter that depends on the velocity. We use a simple exponential decay function that maps velocity values to kappa values:

$$\text{Eq4: } K_v = K_{\max}\left(1+e^{-\gamma(v-v_{\min})}\right)^{-1},$$

where $K_{\max}=100, \gamma=2, v_{\min}=6_{pix/\,frame}$ are constants.

The second term refers to the likelihood of observing an ellipse with orientation β and associated image patch pixel values I given that the true mouse direction is α. Intuitively, the fitted ellipse orientation should either match mouse direction ($\alpha=\beta$) or should be anti-parallel ($\alpha=\beta+\pi$). This can be modeled as a von-Mises mixture model:

$$\text{Eq5: } p(\beta, I \mid \alpha) = w_0 \frac{e^{\kappa\cos(\alpha-\beta)}}{2\pi I_0(\kappa)} + (1-w_0)\frac{e^{\kappa\cos(\alpha-(\beta+\pi))}}{2\pi I_0(\kappa)},$$

where $\kappa=100$ is a constant and w_0 is the mixture coefficient. The weights of the mixtures are determined from the pixel intensity values. We first transform them to a HOG feature vector and then use linear discriminant analysis to reduce its dimensionality to 1D (*x*) by projecting along a direction that best separates head and tail. The exemplars needed to find this projection are collected in the tracking of single mouse videos (see section 2.3). We model $w_0 = p(x \mid \alpha=\beta)$ as a student t-distribution and fit its parameters from the projection n of positive exemplars.

3.4.2 Modeling the state transition matrix

The state transition matrix of the HMM that solves the orientation problem is modeled using the von Mises distribution as a blurred diagonal, representing the fact that if the mouse has a given direction α_t it is more likely to move to a new direction α_{t+1} that is within some standard deviation κ from the current one:

$$\text{Eq6: } p(\alpha_t \to \alpha_{t+1}) = \frac{e^{\kappa\cos(\alpha_{t+1}-\alpha_t)}}{2\pi I_0(\kappa)}$$

4. Correcting mice identities

4.1 State space

Our system uses a Hidden Markov Model (HMM) to associate mouse identities to trajectories. The assignment can be represented as a permutation. For example, [3,1,2,4] -> [A,B,C,D] denotes the assignment of trajectory three to identity A, trajectory

one to identity B, etc. For N mice N! different assignments are possible, therefore the size of the HMM state space is N!. The problem of inferring identities is thus reduced to a sequence in state space. The HMM is solved using the Viterbi algorithm, which finds an optimal state sequence $S=[S_1,S_2,S_3,...,S_T]$ for the given observation sequence $O=[O_1,O_2,O_3,...,O_T]$ by maximizing: $\arg\max_S p(S \mid O,\lambda)$, where λ is the model, defined by its states and state transition matrix.

4.2 Modeling observation probabilities

To propagate information with the Viterbi algorithm we need to define the observation probability: $p(O_j \mid S_j=[i_1,i_2,..i_N])$, i.e., the probability of observing the set of image patches, where each image patch is computed in the fixed size bounding box centered on the detected ellipses, given the assumption that the identity assignment is known (S_j). We assume that the mouse images are independent once the identity of the mouse in each image patch is known, therefore

Eq7: $$p(O_j \mid S_j)=p(O_j^1,O_j^2,...,O_j^N \mid S_j)=\prod_{k=1}^{N} p(O_j^k \mid ID=S_j[k]).$$

That is, the probability of observing the images given state S_j is the multiplication of the probability of each one of the small image patches O_j^k under the assumption that it belongs to identity $S_j[k]$.

To model $p(O_j^k \mid ID=S_j[k])$ we take the pixel values inside image patch O_j^k and transform them to a HOG feature vector. We then reduce the dimensionality to 1D using fisher linear discriminant analysis (LDA), see (Bishop, 2006). Therefore, at the end of this process, we obtain scalar (*x*) which describes O_j^k. The projection coefficients for LDA are computed by setting all the positive exemplars to identity A and all the negative exemplars to mice identities that are not A. Exemplars are collected during the tracking of single mouse videos (see section 2.2). Finally, we model $p(O_j^k \mid ID=S_j[k])$ using location-scale t-distribution:

Eq8: $$p(O_j^k \mid ID=S_j[k]) \sim \frac{1}{\sigma} t\left(\frac{x-\mu}{\sigma},\nu\right),$$

which is fitted to the projection of positive exemplars of identity A. We found t-distribution to give a better fit to the data compared to Normal distribution (Supplementary Fig. 2a-b)

4.3 Modeling state transitions

The ID-assignment state-transition matrix used to solve the identity HMM represents the probability of the transition from one state (an assignment of mouse identities to trajectories) to another from one frame to the next. The mouse identity assignment to two trajectories may only change when the corresponding mice are very close to each other, i.e. the two trajectories are sufficiently close such that their ellipses intersect. We model this constraint by constructing a time-dependent state transition matrix. An entry $a_t(i,j)$ in this matrix represent the probability of switching from state i to state j at frame t.

When all mice are far apart from each other the state transition matrix is set to the identity matrix, representing the condition in which states does not change from one frame to the next. When pair-wise ellipse intersections at frame t are detected at frame t, the corresponding off-diagonal entries of A are set to a value that is different from zero. This signals a non-zero probability that a trajectory swap may take place. For example, suppose that the ellipses of trajectories 2 and 3 intersect. This means that a state of the form [1,*,3,*] can either switch to state [3,*,1,*] or remain in the same state, where * denotes don't care. Rather than estimating the probability of each swap, our system sets all possible swap probabilities to the same value and then normalizes the rows of A to sum to one (row i represents the probabilities of transitioning from state i to all other states).

5. User interface

The graphical user interface (GUI) opens up with a single screen showing a list of all analyzed experiments. An experiment is defined as a collection of videos including both the single mouse videos and the multiple mouse videos. The user can define a new experiment by clicking the “Train” button. The system then asks the user for the single mice videos location and continuous with a fully automated process to track the mouse in each video and train the associated pattern classifier. The color of the “Train” button switches to orange once this process is done. The user can then add long video sequences with multiple mice by clicking “Track”. Videos are automatically sorted by frames timestamp. The system presents the user with the automatically learned background and prompts the user to draw the boundary of the floor of the mouse enclosure (Supplementary Fig. 11c) and attempts to automatically segment mice in 7 random frames with predefined thresholds. The user then verifies the output (Supplementary Fig. 11d) and can correct ellipse placement by moving any one of four control points on the ellipse contour (see Supplementary Fig. 11d inset). Once the user finishes verifying/correcting ellipse placement the system uses this information to

compute the segmentation threshold that offers the best segmentation performance. The system then submits tracking jobs to the computer cluster. Once jobs are finished, the system merges results, corrects for identities and signals the user the results are available by changing the color of the “Track” button to red. The user can then view trajectories overlaid on the video sequences by clicking “Results”.

6 Ellipse distance metric

The weighted distance between two ellipses was defined as:

$$\text{Eq8: } d\left(\theta_1,\theta_2\right)=\sqrt{\frac{1}{D}\sum_{i=1}^{D}\frac{\left(\theta_1^i-\theta_2^i\right)^2}{\sigma_i^2}}$$

Where θ represents ellipse parameters $\left(x,y,a,b,\alpha\right)$ and σ_i^2 is a weight factor (variance) estimated from the difference distributions shown in supplementary figure 6B. The ellipse parameters correspond to the center position of the ellipse (x,y), the major and minor axis lengths (a,b) and the $\left(\alpha\right)$ represents the angle of the major axis and the x axis. Note that the last term $\left(\theta_1^5-\theta_2^5\right)$ is actually computed by taking $\left(\theta_1^5-\theta_2^5\right)\bmod 2\pi$.

7 Validation of ellipse placement

To quantify the performance of the system in placing ellipses on mice two human annotators manually placed ellipses on 455 randomly selected frames (1820 mice images). Four examples of ellipses placed by the human annotators, as well as the automatic segmentation are shown in Supplementary Figure 6a. The second annotator repeated the entire procedure, allowing us to test not only accuracy but also consistency. Histogram of positional, angular and size difference a between annotators are plotted in Supplementary Figure 6b. We found that annotators were consistent in their ellipse placement and that distributions were close to normal.
We defined a metric (see Ellipse distance metric) that allows the comparison between two ellipses in a meaningful way, by taking a weighted sum of the ellipse parameters, where each weight is determined by the standard deviation of the fitted human annotation distributions. A value of 1 in this metric refers to an average distance of one standard deviation along all ellipse dimensions. We plotted the normalized distance metric between the two human annotators and between each human annotators and the automatic segmentation (Supplementary Fig. 6c) and found that most errors are centered around a value of 2, suggesting our automatic segmentation procedure has comparable performance to humans in placing ellipses .

8 Detecting follows using JAABA:

Mouse actions were detected using the Janela Automatic Animal Behavior Annotator (JAABA). Video recordings were made over a continuous 120 hour period for each of the 6 experiments, and then divided into six 1 hour segments. The 'male following' classifier was trained on frames from hours 1 and 12 from exp 5, 1 and 12 from exp 4, and 2 and 12 from exp 1. The training set consisted of 4,875 frames from these segments, with 2,510 frames covering example bouts of following behavior, and 2,365 frames containing negative examples. Only bouts of following initiated by either of the two males in each cage were labeled during classifier training, and the 6 segments were used concurrently to train the classifier.

The accuracy of the classifier was measured by ground truthing its scores on segments that were not used during training, including scores for the 3 cages on different days, and scores for 3 additional cages. Approximately 10,000 frames per hour-long segment were manually scored, and these frames were chosen semi-randomly using an algorithm that selected short segments distributed across each hour, including relatively even numbers of frames that the classifier labeled as "following" or "not following."

The average rates for false alarms and misses for the classifier across all cages and all time intervals ground truthed were 6.3% and 5.6% respectively. By experiment, the average false alarm rates were 6.5%, 4.8%, 5.9%, 8.2%, 6.2%, and 5.8% for experiments1-6. The average rate for misses for experiment were 4.7%, 6.8%, 3.5%, 7.00%, 5.7%, and 5.8% respectively. The mating classifier was trained on 1500 frames and had a false alarm rate of 3.7% and a miss rate of 18.5%.

Supplementary figures legend

Supplementary Figure 1. Segmentation steps. **(a)** Example frame from a single mouse video. **(b)** Automatically learned background model. **(c)** Intensity difference between the frame and the background. **(d)** Binary image is obtained by thresholding the intensity difference image. **(e)** Binary map is closed for holes. **(f)** Fitted ellipse representing mouse pose.

Supplementary Figure 2. Statistical modeling of distributions. **(a)** Histogram of projected HOG features of mouse identity A (blue, positive exemplars) vs. all other mice identities (red, negative exemplars). T distribution fits the data better compared to a Normal distribution. **(b)** Same data plotted as cumulative distributions.

Supplementary Figure 3. Optimal pattern selection. (**a**) Classifier performance for four mice combination as a function of the average false positive and false negative rate.(**b**) Zoom in version of the top 10 combinations. (**c**) Top 10 combinations, broken down to the identities comprising each combination (identities shown below).

Supplementary Figure 4. Two examples of mice burrowing in bedding. Left column, blue mice starts to burrow. Middle column, mouse is completely invisible. Right column, mouse emerges from the bedding with correct identity assignment.

Supplementary Figure 5. Multiple hypotheses initialization. Long movies were split to multiple non-overlapping intervals. To generate the initial mouse placement in the first frame of each interval multiple hypotheses were generated regarding mice position.

Supplementary Figure 6. Quantification of fine positional errors. (**a**) Four examples of ellipses placed by two human annotators (blue and green) and the automatic segmentation (red). (**b**) Differences in position, orientation and size, measured between the two human annotators. Annotator 2 repeated the annotation procedure to measure consistency. (**c**) Accuracy in placing ellipses, as measured by the normalized distance metric (see Supplementary Text). Each curve represents the distribution of distances over all annotated samples between either a human or the machine.

Supplementary Figure 7. Imaging rig.
(**a**) Infrared lights (850 nm) are placed close to the camera, to minimize shadows and provide continuous illumination across the dark/light cycle. (**b**) The video camera (Basler A622f) is fitted with (**c**) an infrared pass filter (pass above 720 nm) which ensures no changes in recorded light levels across the light/dark cycle. (**d**) Square, infrared-transparent tunnels provide shelter without compromising video recording quality. The tunnels are opaque in visible light. (**e**) Mice are bleach marked with individually-distinctive patterns to allow continuous identity tracking. (**f**) Water and (**g**) food are continuously available in multiple locations througout the experiment.

Supplementary Figure 8. Mice favorite places during a five days experiment. Each image denotes a different five day experiment.

Supplementary Figure 9. Dwelling places population analysis. (**a**) Percent of time spent in each of the monitored regions for six experiments, each lasting 5 days. Columns (from left to right) in each experiment represent dominant male, subordinate male and two females. (**b**) Time spent in any of the four corners during dark cycles. Each row correspond to a five day experiment. Color indicates different identities (same conventions as Fig. 5). Each column represent 12 hours.

Supplementary Figure 10. Follow speed and duration distributions (**a**) The distribution of male follow durations for all six experiments. Average follow durations varied slightly across experiments (minimum average: 1.87s, maximum average: 2.52s, Exp 2 duration distribution was significantly different from all curves and Exp 1 was different from Exp 5 at $p<.05$). (**b**) The distribution of male follow speeds for all six experiments. Average follow speeds varied, but not significantly, across experiments (minimum average: 22.6 cm/s, maximum average: 31.1 cm/s, $p>0.05$).

Supplementary Figure 11. Graphical user interface. **(a)** Main menu. **(b)** Main menu after loading video sequences. **(c)** User labels the floor region in the video sequence. **(d)** User corrects automatically placed ellipses for obtaining optimal thresholding parameters. **(e)** Video with overlaid tracking results.

Supplementary Figure 12. Four examples of fight bouts annotation by the software. Identification errors (annotated by a human observer) are denoted by a red X.

References

Bishop, C.M. (2006). Pattern Recognition and Machine Learning.

Felzenszwalb, P.F., Girshick, R.B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. IEEE Trans Pattern Anal Mach Intell *32*, 1627-1645.

Rabiner, L.R. (1989). A tutorial on Hidden Markov Models and selected applications in speech recognition. Proceedings of the IEEE *77*, 257–286